95-702 Distributed Systems
Project 1

Assigned: Friday, January 14, 2011

Due: Friday, January 28, 11:59:59 PM

This project has five objectives:

First, the student is introduced to GlassFish. GlassFish is an open source application server that implements the JEE 5 specification. This tool is used throughout the course. The Netbeans integrated development environment is introduced and is used to build source code and interact with GlassFish.

Second, the student builds his or her first set of distributed systems. The student builds four small web applications using Java Server Pages and servlets.

Third, the student is introduced to the Android simulator. In this simple project, you will be using the simulator's browser capabilities. The simulator runs stand-alone, or within the Eclipse IDE.

Fourth, the student is introduced to mobile device awareness and adapting content to be suitable for either desktop or mobile devices.

And finally, as in all projects this semester, this project requires you to answer a series of reflective questions on Blackboard concerning the solutions you have built. Your answers should clearly and accurately address the non-functional characteristics of your solution, demonstrating a nuanced comprehension of course content and explaining the technical aspects in relation to potential real-world applications. You will be asked to reflect and comment on several important characteristics of distributed systems. For this project, the non-functional characteristics that must be considered include security, protocol layering and interoperability. After completing the project programming tasks, go to Blackboard and answer the questions in the “Project 1 Reflection” assignment.

For each project task, software documentation is required. The software that you write (HTML files, Java files and so on) must contain comments that describe what each significant piece of code is intended to accomplish. Points will be deducted if code is not well documented.

Be sure to consult the rubric linked from the course schedule for details on grading.

For each of the four tasks below, submit a documented servlet and an index.jsp page. The documentation will include your name, a description of each piece of code and well-chosen variable names. There is an example linked on the course schedule showing what we mean by "good documentation".

In addition, you must submit screenshots that demonstrate your programs running. These screenshots will aid the grader in evaluating your project.

Task 1

Write an index.jsp page that requests a user to enter a string of text data. Provide a submit button. When the submit button is pressed a servlet is executed. The servlet must be named ComputeHashes.java. The servlet will compute two cryptographic hash values from the text transmitted by the browser. One hash value will be computed using MD5 and the other using SHA-1. You will need to employ the Java crypto API to compute the MD5 and SHA1 hashes of the text. The original text will be echoed back to the browser along with the two hash values. The hash values will also be sent back to the browser and will appear as hexadecimal text and as text in base 64 notation. We will discuss the use of such hash values later in the course.

To compute the MD5 and SHA-1 hashes, use these standard java packages:

```
import java.security.MessageDigest;
import java.security.NoSuchAlgorithmException;
```

To compute the Bas64 encoding, use the following package:

```
import sun.misc.BASE64Encoder;
```

The BASE64Encoder class is an internal non-documented class. BASE64Encoder objects have a method with the signature String encode(byte[]). It returns a base 64 sting encoding of an array of bytes.

To compute the Hexadecimal representation of a byte array, use the following code:

```
// From the web site "Real's How To"
public String getHexString(byte[] b) throws Exception {
   String result = "";
   for (int i=0; i < b.length; i++) {
      result += Integer.toString((b[i] & 0xff) + 0x100, 16).substring( 1 );
   }
   return result;
}
```

Be sure to provide a user friendly and attractive user interface.

So that you may test your program, an example execution is provided.

```
Computing hashes of Hello of length 5
    SHA-1 (Hex):F7FF9E8B7BB2E09B70935A5D785E0CC5D9D0ABF0
    SHA-1 (Base 64): 9/+ei3uy4Jtwk1pdeF4MxdnQq/A=
    MD5: (Hex): 8B1A9953C4611296A827ABF8C47804D7
    MD5: (Base 64): ixqZU8RhEpaoJ6v4xHgE1w==
```

Task 2

Write a simple web application that allows a user to perform one of three operations on two, possibly very large, integers. The operations will include addition and multiplication as well as an operation to determine if the two integers are relatively prime.

A JSP page will present three input fields to the user. The first two will be used to collect the two integers. The third will be used to collect the operation type. The only operations supported will be "add", "multiply", and "relativelyPrime". Use drop down boxes in XHTML. A submit button will be provided and when it is hit a servlet will be visited. The servlet will be named BigCalc.java and will use the BigInteger class to perform the conversions from strings and the appropriate computation. The servlet will return the result to the browser marked up in HTML. You need to validate both integers and the operation. In the case of invalid input return an error message to the browser - but don't crash the server.

The BigInteger class has multiply and add methods that will be used by the first two operations. For the operation that determines if the two integers are relatively prime use the gcd() method of the BigInteger class. If the greatest common divisor of the two integers is one then the two integers are relatively prime.

Be sure to provide a user friendly and attractive user interface.

Task 3

Write another web application using Netbeans. This application will determine if a string entered into a browser is a palindrome. A string is a palindrome if it is empty, has a single character, or reads the same when reading from left to right or from right to left. Name your servlet Palin.java.

Download and install the Android simulator from Google. Use the browser on the simulator to visit this web application. Produce a screen shot showing the simulator working on your web application.

Notes:

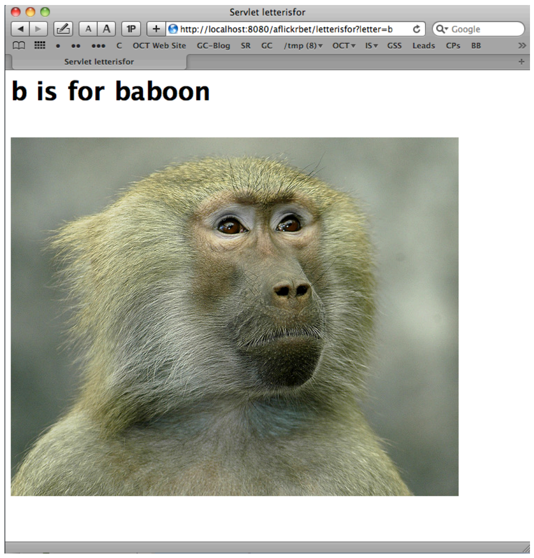
- You will not be able to connect to “localhost” from your Android simulator. One way to solve this is to find the IP address of your machine and use that instead. Another approach is to use Android’s Loopback Address of 10.0.2.2.

Task 4

For task 4, build a web application that will ask the user for a letter, and then reply with the name of an animal that begins with that letter, and a picture of that animal.

In more detail:

1. User is presented with a screen with instructions: “This simple game will map letters to animals, such as "a is for alligator" and will show you a picture of an alligator. Select the letter you would like to play with.
2. User types in a letter a-z, or A-Z, and Submit.
3. If input is valid (for example, “b”) then the user is presented with a reply such as:



b is for baboon

4. This screen should also include “Play again…” and allow the user to type in another letter and submit.
5. The photo should be a link to the Flickr context page for that photo. (In this case, http://www.flickr.com/photos/ucumari/314929277/)

Exceptions:

3a. If the input is not a valid letter, then the user should be presented with a reply that indicates that the input was not a valid letter (e.g. “7 is not a letter, please type in a letter from a-z or A-Z”) and allows them to choose again.

Device awareness and content adaptation:

- Your application should work on both a desktop browser, and an Android browser, and should be device aware and adapt content appropriately.
- Most importantly, you should choose an appropriate photo size for the screen size. A suitably large photo should be used if the user is on a desktop/laptop browser. A suitably small photo should be used if the user is on an Android phone.
- To save bandwidth to the mobile phone, a large photo should not be sent to the browser and just displayed small. Rather, a small photo should be sent to the mobile browser.

Finding photos:

- Photos should be found dynamically from flickr.com. That is, you should not pre-choose the photos to use, but should search for them when needed. (This will keep the game fresh and, occasionally, changing.)
- An example of a useful search to use is:
  http://www.flickr.com/search/?s=int&mt=photos&adv=1&w=getty&m=tags&q=animal+aligator
  This usually provides a list of interesting photos, from the Getty Collection (so some vetting has been done), of the animal "alligator" (as opposed to the company, Everglades Alligator Farm Inc.).
- In the future, retrieving a photo URL from a service like Flickr will be very easy using web services. For now, we will "screenscape" the reply from Flickr. Screenscraping means that your web application will read the HTML reply stream from Flickr and search in it for the information that you want.
- The URLs of the photos, without the rest of the context, has the form:
  http://farm{farm-id}.static.flickr.com/{server-id}/{photo-id}_{photo-secret}_{photo-size}.jpg
- Try the search given in the bullet above, and look at the source of the reply. You will find that by searching for src="http://farm you will find the first picture in the list.
- You can have your servlet do the same search, process the response, and then find the URL of the photograph in the response to use in your application.
- By replacing the {photo-size} letter, you can choose the size of the photo that the URL refers to. Two good sizes to consider are:
  - z – for medium large photographs. (Note: this option is not mentioned in the mashupguide reference listed below.)
    e.g. http://farm1.static.flickr.com/106/314929277_0ac036b9cf_z.jpg

    - m – for small photographs.
      e.g. http://farm1.static.flickr.com/106/314929277_0ac036b9cf_m.jpg
- See http://mashupguide.net/1.0/html/ch02s03.xhtml#d0e1809 for more information.

Mapping of letters to animals:

- It is acceptable to have a fixed mapping of letters to animals. That is, “a” can always be for “alligator”, “b” for “baboon”, etc.
- (In the future, once we have the power of using web services, we may search for alternatives for each letter.)
- (Note: You must search for a picture to correspond to the animal, however, in the way described above. You should not pre-choose the picture.)

Device awareness and content adaptation:

- For this task, you can take a simple approach and just check if the user-agent is an Android device or not.
- Again in the future when we can use web services we can be more sophisticated and search for the capabilities of the device.
- You need to have an appropriate DOCTYPE string (for desktop/laptop or mobile) defined as the first element of your HTML replies.

Naming conventions:
- The Netbeans project should be named “Project1Task4”
- The servlet should be named “LetterIsFor.java”

Summary:

There should be four projects in Netbeans.

Each project will have its own index.jsp

Each project will have a single servlet which would get called from the index.jsp

The Netbeans projects will be named as follows:
- Project1Task1
- Project1Task2

- Project1Task3
- Project1Task4

The servlets will be named as follows:

- Task 1 - ComputeHashes.java
- Task 2 - BigCalc.java
- Task 3 - Palin.java
- Task 4 – LetterIsFor.java

The submission should be a single zip file of all four tasks plus screen shots.